PATENT ABSTRACTS OF JAPAN

(11)Publication number : 60-008944
(43)Date of publication of application : 17.01.1985

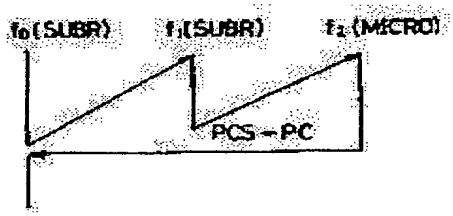
(51)Int.Cl. G06F 9/44

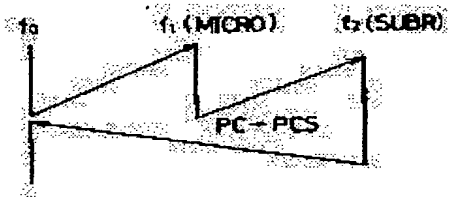
(21)Application number : 58-116525
(22)Date of filing : 28.06.1983

(71)Applicant : FUJITSU LTD
(72)Inventor : HATTORI AKIRA
SHINOKI TAKESHI
KIMURA YASUNORI

(54) TERMINAL RECURSIVE CALL CONTROLLING SYSTEM OF FUNCTION TYPE MACHINE

(57)Abstract:
PURPOSE: To execute a terminal recursive call to each other between a subroutine function and a microroutine function by performing receipt and delivery of a program counter return address.
CONSTITUTION: When a subroutine function SUBRf1 executes a terminal recursive call to a microroutine function MICROf2, a control is delivered to f0 shown by the value of a program counter PC at the time of resetting from f2, by setting in advance the value of a program counter save PCS to the PC. On the other hand, when the MICRO function calls the SUBR function, the SUBR function is started by storing the value of the PC in a PCS area of a stack. That is to say, when executing the terminal recursive call, the value of the PC is stored in the PCS area of a dead own frame, and when resetting from the SUBR function f2, the control is delivered to f0 shown by the MICRO function f1. In this regard, the terminal recursive call means to execute a function call by using suitably a frame of the present call origin function.





LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

⑲ 日本国特許庁（JP）　　⑪ 特許出願公開

⑫ 公開特許公報（A）　　昭60—8944

⑤Int. Cl.⁴	識別記号	庁内整理番号	⑬公開　昭和60年(1985) 1 月17日
G 06 F 9/44		B 7361—5B	

発明の数　1
審査請求　有

（全 6 頁）

⑤④関数型マシンの終端再帰呼出し制御方式

⑳特　　願　昭58—116525
㉒出　　願　昭58(1983) 6 月28日
㉗発 明 者　服部彰
川崎市中原区上小田中1015番地
富士通株式会社内
㉗発 明 者　篠木剛
川崎市中原区上小田中1015番地
富士通株式会社内
㉗発 明 者　木村康則
川崎市中原区上小田中1015番地
富士通株式会社内
㉑出 願 人　富士通株式会社
川崎市中原区上小田中1015番地
㉔代 理 人　弁理士　長谷川文広　　外１名

明　細　書

1. 発明の名称

関数型マシンの終端再帰呼出し制御方式

2. 特許請求の範囲

機械命令とマイクロ命令の実行機能をもつ関数型言語マシンにおいて、機械語で記述されたサブルーチン関数からマイクロ命令で記述されたマイクロルーチン関数を呼出す場合には、機械命令の戻り番地を機械命令のプログラムカウンタに設定して置き、マイクロルーチン関数がサブルーチン関数を呼出す場合には、マイクロプログラムカウンタの値をスタックの所定域へ待避することにより、マイクロルーチン関数が機械命令のプログラムカウンタの管理をすることなしに、サブルーチン関数とマイクロルーチン関数との間で、互いに終端再帰呼出しを実行させることを特徴とする終端再帰呼出し制御方式。

3. 発明の詳細な説明

〔発明の技術分野〕

本発明は、機械命令とマイクロ命令をもつ関数型言語マシンにおいて、関数を呼出し（Call）、復帰（Return）する時のプログラムカウンタ等の制御状態の受渡し方式に関し、特に復帰後に処理をもたない関数からの再帰呼出し、すなわち終端再帰呼出しの制御方式に関する。

〔技術の背景〕

はじめに、本発明の対象である関数型言語マシンについて概述する。関数型言語マシンでは、プログラムは全て関数であり、処理は、必要な関数を順次呼出しながら、実行する形をとる。これらの関数は、機械命令列でできているサブルーチン関数（以下、SUBR関数と呼ぶ）とマイクロ命令列でできているマイクロルーチン関数（以下、MICRO関数と呼ぶ）とに分類される。なお、機械命令は、マイクロ命令に対応させる意味で"マクロ命令"と呼ばれることもある。

一般に、SUBR関数は、ユーザが高級言語で書い

たプログラムを機械語にコンパイルしたものであり、MICRO関数は、マシンに標準的に用意されている組込関数である。高級言語自身は、コンパイルされる以外に、関数型言語専用マシンにおいて、マイクロプログラムでできた高速のインタプリタにより直接実行されることもある。このインタプリタ自身も上記の分類では、MICRO関数に属する。

このようなマシンにおいて、ユーザのプログラムを実行する場合、制御はSUBR関数とMICRO関数の間を行ったり来たりすることになる。

第1図は、上述した関数型言語プログラムの実行を概念的に示したものである。f_0乃至f_3は関数、Callは呼出し、Returnは復帰を表わしている。ここで、たとえば関数f_0，f_1はMICRO関数、f_2，f_3はSUBR関数であることができる。

ところで、マイクロプログラムの実行制御は、基本的にマイクロプログラムカウンタ（以下MPCと略す）に従い、制御状態は基本的にはこのMPCによって規定される。

一方、機械命令プログラムの実行制御は、基本

的に機械命令のプログラムカウンタ（以下PCと略す）に従がい、制御状態は基本的にはこのPCによって規定される。更に、機械命令自身は完全にハードウエアで実現されるマシンもあるが、上記のようなマシンでは、一般にマイクロルーチン（マイクロプログラム）として実現されている。

関数型言語を実行する場合には、通常、各関数ごとにスタックの上にフレームと呼ばれる制御ブロックをつくり、そのフレームに関数名や戻り番地や上位フレームへのリンク等を格納する。関数の処理が全て終了すると、そのフレームはおりたたまれる。

第2図は、その概念図であり、(a)に例示されたf_0，f_1，f_2からなる関数型プログラムを実行したときに、(b)に示すように、呼出し（①，②）ごとに新しいスタックの作成が行なわれ、そして復帰（③，④）ごとに、スタックのおりたたみが行なわれる。

呼出されたMICRO関数では、上記の戻り番地としては、自分の呼出し元関数（Caller）のMPC

値をフレームに格納していればよい。それだけで、自分の仕事が済めば、呼出し元関数に制御を返すことができる。呼出し元関数がSUBR関数であっても、その時のPCをフレームへ格納する必要はない。なぜならば、MICRO関数の実行においては、PCの値を変更することがないからである。

一方、呼出されたSUBR関数では、フレームに格納する戻り番地として、MPCとPCの両方の値が必要である。PCの格納については、SUBR関数がPCを動かして、すなわちPCにしたがって実行されることから、当然に必要である。

SUBR関数の間でのみ制御が授受されている場合には、呼出し元SUBR関数へ復帰する際に、先に格納して置いたPC値から機械命令の実行が開始されることから、MPCを戻り番地としてフレームへ格納する必要はないが、SUBR関数の呼出し元がMICRO関数である場合には、復帰後の手続きを示すMPC値を、スタックフレームへ戻り番地として格納しておく必要がある。

しかし、呼出されたSUBR関数には、呼出し元関

数がSUBR関数であるかMICRO関数であるかを知ることができないから、常にMPCとPCとを戻り番地として格納する必要がある。

このようにして、MICRO関数のフレームは、たとえば第3図に示すような構成をもち、SUBR関数のフレームは第4図に示すような構成をもつ。第3図のMICRO関数のフレームは、1の関数名、2のMPCS（MPC Save域）、3のFPS（Frame Pointer Save域）、および4の実引数からなり、他方、SUBR関数のフレームは、MICRO関数のフレームに5のPCS（Programe Counter Save域）を加えたものである。なお、FPはフレームポインタ、STPはスタックポインタを表わしている。

ところで、MICRO関数が呼出し元へ復帰する時にはそのフレームをおりたたみ、すなわちフレームポインタFPSが指示するフレームへ移り、MPCS域の値をMPCへ戻してそこへ制御を返す。これにより、呼出し元関数の後続手続きが再開される。

これに対し、SUBR関数が呼出し元関数へ復帰する時には、フレームをおりたたみ、PCS域の値を

PCへ戻し、更にMPCS域の値をMPCへ戻してそこへ制御を返す。ここで呼出し元がSUBR関数であった場合には、MPCS値は、PC値が示す機械命令をフェッチして実行を始める手順を指す。しかし、呼出し元がMICRO関数である場合には、MPCS値は、一般に呼出し元関数により定まる後続手続きを指している。

次に、本発明が特にかかわっている終端再帰呼出しについて説明する。

関数型言語においては、前述したように、各関数は他の関数を呼出しながら仕事を進めていくが、最後の関数呼出しの場合には、復帰後に処理が行なわれないため、PCやMPCの待避は不要であるから、呼出すべき関数のフレームを作る必要はない。このため、現在の呼出し元関数のフレームを流用して関数呼出しを行なう。これを、一般に終端再帰呼出し（Tail recursion call）と言う。これにより、フレームの作成、フレームのおりたたみの操作が省略でき、実行速度が向上する。

しかし、第3図に示したように、MICRO関数の

フレームにはPCS域が設けられていないので、MICRO関数を呼出したときに、呼出されたMICRO関数はPCの受渡しを行なわず、また復帰時にPCS域をPCへ戻す操作も行なわない。したがってこのような場合にはSUBR関数からMICRO関数を終端再帰呼出しすると、SUBR関数に制御を戻すことができないという問題があった。つまり、SUBR関数のフレームには、第4図に示したように、そのPCS域にPCの戻り番地が格納されているが、SUBR関数フレームを流用してMICRO関数を呼出すと、呼出されたMICRO関数は、流用されたSUBR関数フレームを第3図に示したMICRO関数フレームと解釈して、復帰時にPCSをPCへ戻さないので、呼出し元SUBR関数へ制御が返らないことになる。

この問題は、MICRO関数にもPCの受渡しをさせるようにすれば一つの解決方法となるが、その反面、通常の呼出しにおけるMICRO関数の処理に無駄な手順が入り、処理が遅くなるという欠点があった。

〔発明の目的および構成〕

本発明の目的は、上記した従来方式の問題を解決することにあり、そのため、MICRO関数は実行時にPCを操作することがないことに着目し、SUBR関数が終端再帰呼出しのため自己のフレームを流用してMICRO関数を呼出したとき、呼出し元SUBR関数の戻りPC値をPCSからPCへ設定、すなわち実質的に待避して置くものである。これにより、呼出されたMICRO関数は、従来方式のように流用スタックのPCS域を管理できなくても、呼出し元SUBR関数へ復帰する際、既に設定されているPCの値を用いて制御を渡すことが可能となる。

本発明の構成は、それにより、機械命令とマイクロ命令の実行機能をもつ関数型言語マシンにおいて、機械語で記述されたサブルーチン関数からマイクロ命令で記述されたマイクロルーチン関数を呼出す場合には、機械命令の戻り番地を機械命令のプログラムカウンタに設定して置き、一方マイクロルーチン関数がサブルーチン関数を呼出す場合には、プログラムカウンタの値をスタックの所定域へ待避することにより、マイクロルーチン

関数が機械命令のプログラムカウンタの管理をすることなしに、サブルーチン関数とマイクロルーチン関数との間で、互いに終端再帰呼出しを実行させることを特徴としている。

〔発明の実施例〕

以下に、本発明の詳細を実施例にしたがって説明する。

本発明方式の原理を用いた場合、SUBR関数およびMICRO関数間での呼出し制御は、次のように行なうことができる。

基本的には、SUBR関数が制御をもっている時には、その戻りPC番地はスタックフレームのPCS域へ格納しておくが、MICRO関数が制御をもっている時には、戻りPC番地をPCに設定して置くようにする。つまり、SUBR関数がMICRO関数を呼出す時には、戻りPC番地をPCへ放置してMICRO関数の起動をかける。さらに詳しく言うと、終端再帰呼出しの時には、自フレームのPCS域をPCへ移し、非終端再帰呼出しの時には、CALL命令の次のPCアドレスをそのままPCへ置いておくようにする。

第5図(a)は前者の1例を示し、SUBR関数f_1がMICRO関数f_2を終端再帰呼出したとき、PCSの値をPCに設定して置くことにより、f_2からの復帰時にPCの値が示すf_0へ制御を渡すことができる。第5図(b)は後者の1例を示し、SUBR関数f_1がMICRO関数f_2を非終端再帰呼出した場合、現PC値の次の番地を戻り番地としてPCに設定して置けば、f_2からの復帰時に、PCが示すf_1の戻り番地からf_1を再開することができる。

一方、MICRO関数がSUBR関数を呼出す時には、PCの値をスタックのPCS域へ格納して、SUBR関数の起動をかける。詳しく言うと、終端再帰呼出しの時には、第5図(a)に示すように自フレームのPCS域（MICRO関数だから空いている。）へ、PCの値を格納し、非終端再帰呼出しの時には、第5図(d)に示すように呼出すべきSUBR関数のために作ったフレームのPCS域へPCの値を格納する。

以上のことから、関数呼出しの際のPCの受渡し方式は以下のようになる。なお、◎印は、SUBR関数とMICRO関数との間で制御が移動するものである。

・ MICRO関数を非終端再帰呼出しするとき
〔呼出されるMICRO関数フレームのPCS域は空で呼出し
◎ SUBR関数を非終端再帰呼出しするとき
〔PCを呼出されるSUBR関数フレームのPCS域へ格納〕してから呼出し

第6図は、以上のPC受渡し方式を適用した関数呼出し制御の例であり、フレームのPCS域とPCの設定規則を示している。図中、実線の矢印は、終端再帰呼出しの場合の制御データの流れ、そして点線の矢印は、非終端再帰呼出しの場合の制御データの流れを表わしている。なお、本図におけるように、SUBR関数を呼出すMICRO関数は特定の関数（インタプリタ関数）だけである。

〔発明の効果〕

以上のように、PC戻り番地を受渡すことにより、MICRO関数側がスタックフレームにPCを格納したり、PCへ戻したりすることなく、SUBR関数とMICRO関数の間で互いに、終端再帰呼出しすることが可能になり、関数型言語の処理効率を高めることができる。

また、前述したように、終端再帰呼出しのときは呼出し元フレームを流用し、非終端再帰呼出しのときには新フレームを作成するものとする。

(1) 呼出し元がSUBR関数の場合
◎ MICRO関数を終端再帰呼出しするとき、
〔自フレームのPCS域→PC〕してから呼出し
・ SUBR関数を終端再帰呼出しするとき、
〔自フレームのPCS域 そのまま〕で呼出し
◎ MICRO関数を非終端再帰呼出しするとき
〔PCはそのまま、かつMICRO関数フレームのPCS域は空〕で呼出し
・ SUBR関数を非終端再帰呼出しするとき
〔PCを呼出されるSUBR関数フレームのPCS域へ格納〕して呼出し

(2) 呼出し元がMICRO関数の場合
・ MICRO関数を終端再帰呼出しするとき
〔自フレームのPCS域は空〕で呼出し
◎ SUBR関数を終端再帰呼出しするとき
〔PC→PCS域〕してから呼出し

4. 図面の簡単な説明

第1図は関数型言語プログラムの概念図、第2図(a)，(b)はフレームの作成およびおりたたみの説明図、第3図はMICRO関数フレームの構成図、第4図はSUBR関数フレームの構成図、第5図(a)乃至(d)は本発明方式の原理説明図、第6図は制御データ受渡しの実施例説明図である。

図中、f_0，f_1，f_2，f_3は関数型言語の関数、2はMPCS（マイクロプログラム・カウンタ・セーブ）域、3はFPS（フレーム・ポインタ・セーブ）域、5はPCS（プログラム・カウンタ・セーブ）域、FPはフレーム・ポインタ、STPはスタック・ポインタを表わす。

特許出願人　富士通株式会社
代理人弁理士　長谷川文廣（外1名）

第1図

第2図

第3図

第4図

第5図

オ 6 図